MANIPULATING POLYGONS

REBECCA HOBERG AND MARY SOLBRIG

Abstract. We will implement various algorithms on polygons (not necessarily convex). The user will input an ordered list of vertices, or a list of such lists, and our program then initializes a corresponding list of polygons. If the vertices the user gives have adjacent colinear points, these will be removed, and if they have self-intersections, more than one polygon may be created. The user also has the option to specify holes inside the polygons. Functions we will create include contains (whether a point or polygon is contained in a polygon) intersection of two polygons.

Contents

1. Segment Class

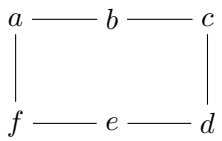
In order to work more easily with polygons, we first created a class called segment. In this class is the function intersect. The call intersect(segment1,segment2) returns (True, point) if the two segments intersect at a point (including possibly the endpoints), and (False, None) if they do not intersect. If their intersection is in fact a line segment, it returns (True, None). The segment class also has contains, which determines whether a point is contained in the segment, and length.

2. Polygons Class

The class polygon_set that we implemented has a list of polygons called corners, where each polygon in the list is a list of vertices that ends with its starting vertex.

Date: June 10, 2013.

Here is an example of how our initialization works. Suppose you input the sequence of points $[a, b, c, d, e, f]$, corresponding to the following diagram.



Our code would then set the polygons list to be $[[a, c, d, f, a]]$.

If on the same diagram we were instead given the list $[a, b, e, d, c, e, f]$, our code would set the polygons list to be $[[a, b, e, f, a], [c, e, d, c]]$.

The class polygon_set will also have the following functions:

(1) Intersection
(2) Area
(3) Contains
(4) Perimeter

3. Implementation Details

The initialization of polygons calls the function set_corners, which does the following:

(1) Removes any colinear points.
(2) If any segment intersects a non-adjacent segment, it computes the point of intersection and splits the list into two separate lists of vertices.

IMPORTANT: If the list of vertices the user gives would yield interior line segments, our code will (at best) fail loudly, and (at worst) fail silently. For example, a star will fail. Hopefully will fix this by tomorrow, but may not get around to it.

4. What is currently in Sage

Sage currently has some polygon capabilities, but most of it is done in much greater generality, considering polyhedra in n dimensions. While we were able to find the simple area function, we weren't able to find union and intersection and it only deals with convex polygons. Our code allows the user to compute the area of much more general 2-D polygons, and also allows the computation of intersection. Union of two polygons would also be fairly simple to create, as it would be fairly similar to intersection.

An alternative Sage object, polygon2d, allows for arbitrary polygons, even allowing for self-intersections, but all it does is graph. Our code uses this object to be able to graph, but it also allows for computation of intersection, union, and more.